

12/31/98



12/31/98 U.S. PTO

Attorney Docket No. 42390.P2319R

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Assistant Commissioner for Patents  
Washington, D.C. 20231

REISSUE APPLICATION TRANSMITTAL

Transmitted herewith is the application for reissue of U.S. Patent No. 5,590,342 issued on December 31, 1996.



Inventor(s): Suresh K. Marisetty

Title: METHOD AND APPARATUS FOR REDUCING POWER CONSUMPTION  
IN A COMPUTER SYSTEM USING VIRTUAL DEVICE DRIVERS

Enclosed are the following:

1. Specification, Claim(s) and Drawings(s)

(a) 20 page(s) of specification, claims and/or abstract

(b) X 7 sheet(s) of drawings(s)  
X formal  
informal

No changes in the drawings upon which the original patent was issued are to be made. Therefore, in accordance with 37 CFR 1.174, please find attached, in the size required for original drawings:

X a copy of the printed drawings of the patent.  
a photoprint of the original drawings.

2. Declaration and Power of Attorney

X 4 page(s) of declaration and power of attorney (unsigned)

3. Preliminary Amendment (check if applicable)

           attached

4. Offer to Surrender the Original Letters Patent in Accordance With 37 CFR 1.178 is attached

X Offer to surrender is by the inventor (unsigned)  
X along with assent of assignee (unsigned)  
           offer to surrender is by the assignee of the entire interest (and the reissue application does not seek to enlarge the claims of the original patent).

5. Letters Patent

           original letters patent attached  
           declaration that original letters patent lost or inaccessible

6. Title

In accordance with 37 CFR 1.171 this application for reissue is accompanied by

\_\_\_\_\_ a certified copy of an abstract of title  
or  
X an order for an abstract of title

7. Information Disclosure Statement (check if applicable)

\_\_\_\_\_ attached

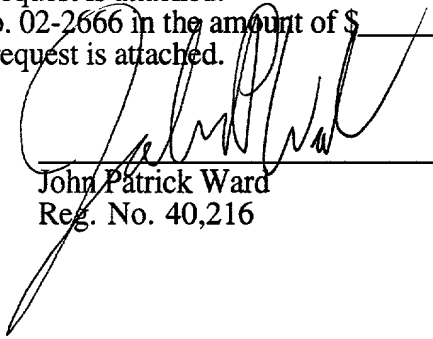
8. Fee Calculation (37 CFR 1.16(h), (i) and (j))

CLAIMS AS FILED					
	Number Filed	Number Extra	Rate	Basic Fee	\$ 760
Total Claims	50	30 (and also in excess of total claims in patent)	x \$18		\$ 540
Independent Claims	7	4 (number of independent claims in patent)	x \$78		\$ 312
Filing Fee Calculation					\$1,612

9. Method of Payment of Fees

X Enclosed is a check in the amount of \$ 1,612.00.  
X Enclosed is a check in the amount of \$ 25.00.  
X Charge any shortages or credit any coverages to Account No. 02-2666.  
 A duplicate of this request is attached.  
 \_\_\_\_\_ Charge Account No. 02-2666 in the amount of \$ \_\_\_\_\_.  
 A duplicate of this request is attached.

Date: 12/31/98

  
 John Patrick Ward  
 Reg. No. 40,216

12400 Wilshire Boulevard  
 7th Floor  
 Los Angeles, California 90025-1026  
 (408) 720-8598

"Express Mail" mailing label number: ELO34147231US  
 Date of Deposit: December 31, 1998

I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231

Lisa Kaiser  
 (Typed or printed name of person mailing paper or fee)

Lisa Kaiser  
 (Signature of person mailing paper or fee)

12/31/98  
 (Date signed)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Reissue Application No.:

Filed: Concurrently herewith



Patent No.: 5,590,342

Granted: December 31, 1996

Patentee: Suresh K. Marisetty

Title: METHOD AND APPARATUS FOR REDUCING POWER CONSUMPTION IN A COMPUTER SYSTEM USING VIRTUAL DEVICE DRIVERS

Assistant Commissioner for Patent  
Washington, D.C. 20231

REQUEST FOR ABSTRACT OF TITLE

1. Please prepare a certified Abstract of Title in respect of the above identified original patent for placing in the official file of the Reissue Application which is filed herewith.

2. (Complete the following, if applicable)

Title in the name of INTEL CORPORATION  
was recorded on January 24, 1995, Reel 7341, Frame 694.

3. For the fee required by 37 CFR 1.19(b)(4) please

- ☒ find enclosed a check in the amount of \$25.00.  
☒ charge any shortages or credit any overages to Account No. 02-2666.  
 A duplicate of this request is attached.  
☐ charge Account No. 02-2666 in the amount of \$15.00.  
 A duplicate of this request is attached.

Date: 12/31/97

John Patrick Ward  
Reg. No. 40,216

12400 Wilshire Boulevard  
7th Floor  
Los Angeles, California 90025-1026  
(408) 720-8598

"Express Mail" mailing label number: EL03414023141  
 Date of Deposit: Dec 31, 1997  
 I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231  
Lisa Kaiser  
 (Typed or printed name of person mailing paper or fee)  
Lisa Kaiser  
 (Signature of person mailing paper or fee)  
12/31/98  
 (Date signed)

UNITED STATES PATENT APPLICATION

for

METHOD AND APPARATUS FOR REDUCING POWER CONSUMPTION IN  
A COMPUTER SYSTEM USING VIRTUAL DEVICE DRIVERS

Inventor:

Suresh K. Marisetty

prepared by:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN

12400 Wilshire Boulevard

Los Angeles, CA 90025-1026

(408) 720-8598

File No.: 42390.P2319

EXPRESS MAIL CERTIFICATE OF MAILING

"Express Mail" mailing label number TB377895705

Date of Deposit November 29, 1994

I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to the Commissioner of Patents and Trademarks, Washington, D.C. 20231.

Triva Lewis

(Typed or printed name of person mailing paper or fee)



(Signature of person mailing paper or fee)

1  
**METHOD AND APPARATUS FOR  
REDUCING POWER CONSUMPTION IN A  
COMPUTER SYSTEM USING VIRTUAL  
DEVICE DRIVERS**

5

**FIELD OF THE INVENTION**

The present invention relates to the field of computer systems; more particularly, the present invention relates to reducing power consumption in a computer system using device drivers.

**BACKGROUND OF THE INVENTION**

Typically, a computer system contains a processor, a bus, and other peripheral devices. The processor is responsible for executing instructions using data in the computer system. The bus is used by the processor and the peripheral devices for transferring information between one another. The information on the bus usually includes data, address and control signals. The peripheral devices comprise storage devices, input/output (I/O) devices, etc. Generally, all operations being performed in the computer system occur at the same frequency.

Many of today's computer systems include power management capabilities. Power management is used to reduce the dynamic and static power consumption of a system to increase the battery life of a mobile personal computer (PC) or to reduce the energy costs associated with a desktop PC. Dynamic power is consumed by all components during state switching of internal electronic circuits, while static power is consumed due to the leakage currents of electronic devices.

The existing power-management techniques in a typical notebook and desktop PC use specific hardware mechanisms to provide maximum power savings. These hardware mechanisms use processor specific interrupts (e.g., System Management Interrupt (SMI)) and other system activity monitoring hardware (e.g., idle timers and hardware trapping mechanisms) to provide a reasonable amount of power conservation.

Alternatively, some software mechanisms exist, which are used today to detect CPU idle conditions in order to put the system in an optimum power conservation mode (e.g., Windows APM driver, DOS POWER.EXE, etc.). Although the available software techniques provide for about seventy to eighty percent of power conservation, they do not power manage a system beyond CPU idleness. That is, they do not detect idleness of I/O devices nor turn off idle I/O devices during system operation or slow the CPU clock rate, etc.

In existing power management architecture's, there are several dynamic and static power conservation states. One of these states is referred to as a fully on or full power on state, in which all the components of a typical system are powered. In this state, all the clocks in the system will be running at full speed. This state offers no power savings. Another state is referred to as a local stand-by, or partially powered-on, state, in which certain temporarily idle local devices in the system, such as a floppy device, graphics devices (e.g., LCD, CRT), hard disk device, etc. are powered down. The power to these turned off devices is restored when an internal or external system event requires the services of these resources. The system maintains idle timers for each of these power-manageable devices. The idle timers enter an expired time-out state when they detect idleness of these devices after a pre-defined period of inactivity, and notifies

the power management software. This state offers the minimal amount of power savings in a system. Another state is referred to as a global stand-by state, in which most of the system devices are powered down with the exception of the CPU and the system DRAM memory. The clock to the CPU is stopped with the DRAM memory operating in an extended power conservation mode, sometimes referred to as stand-by mode with self refresh. At this point in time, the CPU and DRAM are ready to be activated when a system event occurs. An example of such a system event is a keyboard/mouse click or other system interrupts (e.g., IRQ0-IRQ15, NMI, SMI, etc.). The last power management state is referred to as hibernation, where the system is put in the power-off state. When a system detects an idle condition, after a predetermined period of time in the global stand-by mode, it can initiate a transfer to the hibernation state. In such a state, complete system state is saved to the hard disk. When the system is turned back on, the hibernation state restores the system back to exactly the same state as it was before.

Dynamic clock throttling is the state where the dynamic power consumed by a CPU is reduced by slowing its clock rate. A slow clock to the CPU is emulated by periodic assertion and de-assertion of a stpclk (stop clock) signal. This slow clock emulation leads to less power consumption by the overall system. This mode is activated during normal operation of a system and it is overlapped with a fully-on state to offer additional power savings during the fully-on state.

As shown above, the prior art system of power management requires the detection of local and global system events. This is typically handled by special hardware or power-aware applications and device drivers. The detection of system idleness for global stand-by is accomplished by monitoring their interrupt activity in software or hardware using existing methods. If none of the system interrupts are activated in a predetermined of time, a global stand-by event is generated by the chosen hardware or software mechanism. Local activity of individual I/O devices is detected mostly by dedicated power management hardware. The hardware snoops on I/O device resources (e.g., I/O addresses and IRQx, etc.). The mapping of the resources is static and deterministic and known at system boot-up time. These I/O resource mappings do not change over the lifetime of the current system boot. In certain power management implementations, the snoop I/O addresses are programmable in the I/O hardware, while they are fixed in other systems. The deterministic nature of the mappings of these I/O resources of the local devices (as per PC-AT/DOS standards) makes it easy to design standard hardware which is consistent across all PC DOS platforms.

These described methodologies have several inherent problems. For instance, each of the I/O devices needs an idle timer to monitor the activity. This imposes a restriction on a number of hardware timers that can be designed into the system. Also, most implementations hardcode the I/O trapping address of the I/O devices to save "gates". This makes a system more sensitive to remapping of the I/O resources. Furthermore, the existing mechanisms assume that all I/O devices use standard I/O resource mappings over the lifetime of the system, i.e., static I/O and IRQx mapping. This, in fact, places a severe restriction on the usage of the system resources and demands perfect hardware compatibility across all platforms.

Power management software in the traditional system is completely decoupled from the operating system and application. This makes the system prone to the operating system

and power management software performing activities, with neither of them being aware of the activities being performed by the other. This may lead to system crashes where a power management interrupt, such as SMI, takes control away from the operating system while it is executing in the middle of a critical section of code. 5

The current generation of device drivers in operating systems virtualize I/O ports. When the I/O ports are virtualized, it becomes difficult, and in some cases impossible, for the power management software and hardware to detect any possible remappings. This leads the power management hardware to monitor and trap on invalid I/O device addresses, thereby generating improper events in the system. 10

In a plug-and-play environment, it is assumed that the I/O device resource mappings (I/O and IRQx) are no longer deterministic or visible to the power management software at system boot-up time. Current and future generations of operating systems will be based on plug-and-play architectures, where the I/O resource mapping can and will change dynamically during the lifetime of the current system boot. When these dynamic remappings of I/O devices do occur, there is currently no easy way to communicate to the power management hardware and software. 15 20

Also, the existing software power-management techniques assume that the applications of the associated device drivers in the system are APM and PMC aware. This makes it difficult to manage a system with applications and drivers which are not power aware. 25

#### SUMMARY OF THE INVENTION 30

A power management mechanism for use in a computer system is described. The computer system comprises a bus, a memory for storing data and instructions, and a central processing unit (CPU). The CPU runs an operating system having a power management virtual device driver (PMVxD) responsible for performing idle detection for devices. The PMVxD performs idle detection using event timers that provide an indicator as to the activity level. The PMVxD places idle local devices in a reduced power consumption state when no activity has occurred for a predetermined period of time. 35 40

#### BRIEF DESCRIPTION OF THE DRAWINGS 45

The present invention will be understood more fully from the detailed description given below and from the accompanying drawings of various embodiments of the invention, which, however, should not be taken to limit the invention to the specific embodiments, but are for explanation and understanding only. 50

FIG. 1 illustrates one embodiment of the power management architecture of the present invention.

FIG. 2 illustrates one embodiment of the power management states of the present invention. 55

FIG. 3 illustrates an embodiment of the power management control of the present invention.

FIG. 4 illustrates one embodiment of the dynamic clock throttling mechanism of the present invention. 60

FIG. 5 illustrates a timing diagram of the CPU clock throttling.

FIG. 6 a conceptual diagram of the Windows operating system in enhanced mode. 65

FIG. 7 is a block diagram of one embodiment of the computer system of the present invention.

DETAILED DESCRIPTION OF THE PRESENT  
INVENTION

A method and apparatus for reducing power consumption in a computer system is described. In the following detailed description of the present invention numerous specific details are set forth, such as types of I/O devices, idle time periods, interrupt types, power management states, etc., in order to provide a thorough understanding of the present invention. However, it will be appreciated by one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form, rather than in detail, in order to avoid obscuring the present invention.

Some portions of the detailed descriptions which follow are presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like.

It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities. Unless specifically stated otherwise as apparent from the following discussions, it is appreciated that throughout the present invention, discussions utilizing terms such as "processing" or "computing" or "calculating" or "determining" or "displaying" or the like, refer to the action and processes of a computer system, or similar electronic computing device, that manipulates and transforms data represented as physical (electronic) quantities within the computer system's registers and memories into other data similarly represented as physical quantities within the computer system memories or registers or other such information storage, transmission or display devices.

The present invention also relates to apparatus for performing the operations herein. This apparatus may be specially constructed for the required purposes, or it may comprise a general purpose computer selectively activated or reconfigured by a computer program stored in the computer. The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general purpose machines may be used with programs in accordance with the teachings herein, or it may prove convenient to construct more specialized apparatus to perform the required method steps. The required structure for a variety of these machines will appear from the description below. In addition, the present invention is not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the invention as described herein.

## Overview of the Present Invention

The present invention provides for power management in a computer system using virtual device drivers (VxDs). In one embodiment, the VxDs of the present invention are



provided in the Windows<sup>TM</sup> operating system manufactured by Microsoft® Corporation of Redmond, Wash. The present invention provides a power management VxD that controls at least a portion of the power management in the computer system. Power management in the present invention is facilitated by the I/O/interrupt/VxD trapping capabilities of the VxDs running in the protected mode of the CPU at the highest privileged level of the operating system in an operating system co-operative manner. The VxD operates at the CPU's highest privileged level (ring 0). Therefore, the VxD of the present invention has access and control over system hardware and software components. This enables it to operate and respond to power management events much faster, with low task switching overheads. In one embodiment, not only does the VxD interface to other software (e.g., applications), but it also provides an API interface to both real/protected mode programs.

The present invention allows software to power-manage a system with applications and device drivers which are not power-aware. In addition, software operates cooperatively with the existing power-management software infrastructure such as, for instance, Advanced Power Management specification (APM 1.1), Power Management Coordinator (PMC), etc.

#### The Windows Environment

The Microsoft® Windows operating environment provides a graphical user interface (GUI) which makes Windows application programs easier to use. Microsoft® Windows environment runs Windows applications located in the extended area of memory above the 1 megabyte boundary using the protected mode of a processor, such as an Intel Architecture Microprocessor, manufactured by Intel Corporation of Santa Clara, Calif., the corporate assignee of the present invention.

The Microsoft® Windows 3.1 system can operate in one of two modes: "standard" mode and "enhanced" mode. The standard mode exists so that personal computers equipped with older 80286 processors can use the Windows environment. The enhanced mode of Microsoft® Windows is used when Microsoft® Windows is run on a computer system which uses an 80386 microprocessor or more recently available microprocessors such as, for instance, the i486<sup>TM</sup> processor.

The enhanced mode of Microsoft® Windows operates in the protected mode of the Intel Architecture Microprocessors (e.g., i386<sup>TM</sup> processor, i486<sup>TM</sup> processor, etc.). In this manner, the enhanced mode of Microsoft® Windows takes advantage of features in the Intel Architecture Microprocessors to offer virtual memory and multitasking operation. The processor hardware supports execution of several Windows applications in protected mode.

The enhanced mode of Microsoft® Windows supports DOS applications using "DOS virtual machines." In a DOS Virtual Machine, the Intel Architecture Microprocessor operates in Virtual 8086 mode and uses the virtual memory feature to provide DOS, device drivers, and Terminate and Stay Resident (TSR) programs originally loaded into the computer to a DOS virtual machine in extended memory. Windows uses the virtual memory system to make the application area and the DOS, device drivers, and TSR programs appear to be a single contiguous block of real mode memory. When the microprocessor is operating in Virtual 8086 mode within the address area of DOS virtual machine, the Virtual 8086 mode microprocessor is unaware of any memory outside of the DOS virtual machine.

FIG. 6 provides a conceptual diagram of the Windows system in enhanced mode. Referring to FIG. 6, the computer

system hardware is represented as a level. The Ring 0 level with the Kernel and virtual device drivers (VxDs) is also shown, along with the system virtual machine and the various DOS virtual machines. Windows creates DOS virtual machines by mapping DOS, device drivers, and TSR programs in the system VM into the DOS VMs. Therefore, all the virtual machines share a region of memory called the shared real mode memory.

The Virtual device drivers (VxDs) at ring 0 are a special feature of Microsoft® Windows enhanced mode. A virtual device driver is actually a routine which manages a system resource such that more than one application can use the system resource at a time. Virtual device drivers therefore support Windows' ability to act as a multitasking operating system. The virtual device drivers, including the VxD of the present invention, have access to a wide range of kernel services, including those for hardware management, memory management, task scheduling, and communicating with other virtual devices.

As illustrated in FIG. 6, all the Windows applications run within the system virtual machine which operates in protected mode. The Windows Dynamic Link Libraries (DLLs) which support Windows applications also run within the system virtual machine.

Each DOS application in FIG. 6 runs within its own DOS virtual machine. Since the DOS virtual machines usually operate in the Virtual 8086 mode of the microprocessor, the DOS applications generally only address the 1 Megabyte of memory in the DOS virtual machine.

#### Power Management Architecture

The power management architecture of the present invention is shown in FIG. 1. Referring to FIG. 1, various power-aware applications (101, 102) shown make use of an APM/PMC device driver 103 at the operating system software layer. A power-aware device driver 106 also makes use of APM/PMC device driver 103 at the operating system software layer. The APM BIOS (Basic Input/Output System) hardware 104 controls the APM/PMC device driver 103. The APM BIOS 104 also controls hardware 105.

Also shown in FIG. 1, power-unaware applications 110-112 communicate with PMVxD power management software 113. The PMVxD power management software 113 communicates with a Plug-n-Play (P-n-P) configuration manager 114 and the APM/PMC device driver 103 in the operating system software layer. The PMVxD power management software 113 controls hardware. The PMVxD power management software 113 controls the VxD controlled hardware 115 and 116 as well as the power management hardware 117 at the hardware layer. In one embodiment, the VxD controlled hardware 115 and 116 have built-in power management capabilities in the form of a switch and only needs some type of signal to enable them. In one embodiment, the power management hardware 117 may include hardware necessary for placing certain I/O devices in a reduced power consumption state.

The present invention provides for placing a computer system in various power management states, or modes of operation, using a virtual device driver (VxD) that has I/O, device driver and interrupt trapping capabilities. In one embodiment, the VxD provides support for four power management modes: fully-on, local standby, global standby and hibernation. The PMVxD of the present invention also provides support for clock throttling. The present invention may support, only one or more of these modes. Furthermore, the use of the clock throttling mode depends on whether the processor in a computer system includes the required functionality to provide such a feature (i.e., repeatedly issuing a

HALT command to the CPU). One embodiment of the power management states of the present invention are illustrated in the state diagram in FIG. 2.

Referring to FIG. 2, state 1 is the fully powered-on state. While the system is active, the computer system remains in the fully powered-on state. Whether the system is active is based on device activity of local devices as well as system activity (e.g., keyboard stroke, mouse movement, etc.) Once one or more local devices are determined to be idle, the computer system transitions to state 2 where each idle local device is powered off, such that the computer system enters the local standby state with respect to those powered down devices. The computer system remains in local standby while a local device is idle. As soon as the local device is no longer idle, the computer system transitions back to the fully powered on state. When the computer system determines that the entire system is idle (e.g., all the local devices are idle), the computer system transitions to state 2 to enter global standby. In global standby, the system is placed in sleep mode, where it remains until system activity is detected. At that time, the computer system returns to the fully powered on state (0).

In another embodiment, the system transitions from the local standby state directly to the global standby state. That is, the computer system does not reenter the fully powered on state before entering global standby.

The Power Management Virtual Device Driver (PMVxD)

The present invention comprises a power management virtual device driver (PMVxD) and a set of data structures. The data structures are initialized at system boot-up time to provide command/status information to the PMVxD. The PMVxD controls power management and comprises several software idle timers, one for each enabled I/O device. The idleness of a particular device is detected by monitoring the activity of each of the enabled I/O devices. The monitoring of the enabled I/O devices may be performed by one of the following: I/O port address trapping, chaining into I/O device interrupt handlers, trapping on I/O devices driver (VxD) accesses, and chaining into I/O protection fault handler, each of which is well-known in the art.

Most of these capabilities are provided to the VxD as a set of VMM and VxD service calls, which are standard Windows™ device driver support.

The PMVxD is chained into the system timer interrupt, which provides the time base for all PMVxD counters. In one embodiment, the PMVxD uses the occurrence of the IRQ0 to indicate when to monitor the activity of each I/O device and also the overall system activity for local and global standby modes. In one embodiment, the IRQ0 occurs every 55 ms. Thus, in this embodiment, a power management manager of the PMVxD checks every 55 ms to determine whether the local devices have been or are active and whether the system as a whole has been or is active. If a device has been inactive for a predetermined period of time, the power management manager of the PMVxD controls the powering down of the device. The predetermined period of time may be of variable length and is set based on the desired power savings. Different periods of time may be associated with different devices.

The PMVxD installs handlers for I/O trapping to the I/O port of each device. Anytime an I/O port access is trapped, a corresponding counter is updated (increment/decrement) to reflect the activity status. For I/O devices whose I/O addresses are virtualized, PMVxD interrupt handler stubs (e.g., small pieces of code stored in memory at all times) can be chained into the original interrupt handlers for each specific I/O device. This PMVxD stub handler maintains the

idleness of an I/O device, for example, Floppy Disk interrupt OEH. The use and operation of a stub handler is well-known in the art.

FIG. 3 illustrates the power management control overview. Referring to FIG. 3, the PMVxD 301 is shown controlling hardware 302. Software 303 also controls hardware 302 and places hardware 302 in hibernation mode in cooperation with hibernation timer 304. The PMVxD 301 comprises software timers, including a system events timer 301A and multiple local events timers, such as a floppy events timer 301B, graphics events timer 301C and I/O device timer 301D. Each of the local events timers (e.g., 301B-D) monitor local events via an I/O trap handler, a device driver hook handler or a chained-interrupt trap handler as an interface. This interface increments and decrements the software timers. The system events timer 301A monitors global events via an interface, which may also be either an I/O trap handler, a device driver hook handler or a chained-interrupt trap handler. Note that in one embodiment only one of these is employed for each global or local events timer.

At a regular interval, such as at every 55 ms corresponding to the occurrence of the IRQ0, the power management (PM) manager 301E checks the status of each of the events timers (e.g., 301A-D). When an events timer times-out, the PMVxD may turn off the power to the I/O device. That is, PM manager 301E causes a handler for that event timer to be called. This handler controls the dedicated hardware in hardware 302 for removing power to the I/O device. When activity is detected, PM manager 301E calls the handler again to power up the device. Similarly if the system events timer times out, PM manager 103E calls the global standby handler associated with the system events timer, which when run causes the clock to the CPU to be stopped in a manner well-known in the art, such that the CPU is halted.

When the CPU has been halted, the hibernation timer 304 is started. If the hibernation timer times out, an interrupt occurs, such as a system management interrupt (SMI) in one embodiment. Software 303 for the interrupt places the system in a hibernated state. In response to a system event such as, for instance, a keyboard input or cursor control device movement, the computer system exits the hibernation state and the global standby state.

PMVxD 301 also includes a slow clock timer 310 for use with the clock throttling mode. The slow clock timer 310 is described below.

With respect to Plug-and-Play Compliance, the PMVxD is part of the O/S and appears as a device driver in the system. In a Plug-and-play environment, when the system resources are remapped to the I/O devices, the PMVxD is informed of the changes by the O/S specific configuration manager or resource manager. This well-defined interface mechanism between the O/S and PMVxD allows it to dynamically adapt itself to the changes gracefully. The present invention provides such capability by having the PMVxD register with the configuration manager (CM) and instructs the configuration manager to notify it when there has been a configuration change. The PMVxD responds to the notification in the same manner as when examining the data structures at system boot-up time.

#### Dynamic Clock Throttling

FIG. 4 illustrates the dynamic clock throttling mechanism of the present invention. Referring to FIG. 4, dynamic clock throttling is accomplished by the PMVxD operating in the Windows 3.1 O/S 400 with support from the standard PC hardware. Periodic assertion of CPU HALT to CPU 404 by the slow clock timer 402 emulates a slower clock timer

event. CPU 404 being responsive to the HALT command causes a reduction in power consumption. FIG. 5 illustrates a timing diagram of the CPU clock throttling. Using speed up timers 401, the slow clock timer 402 is de-asserted by a variable and periodic interrupt event from the system timer 405 or real-time clock (RTC) 406.

#### Data Structures of the PMVxD

The following tables illustrate one embodiment of the data structures of a PMVxD according to the teachings of the present invention. These data structures have a one-to-one correspondence to the programmable hardware power management features of the i486SL SuperSet. In one embodiment, a graphical user interface (GUI) permits changes to be made to default values as per the needs of an end user. Tables 1 and 2 below illustrates the PMVxD initialization data structures of local devices.

TABLE 1

Struct Local_Device	Struct Local_DEvents	Struct Local_DStatus
<pre> { IntFloppyEn/Di; Int 1 larddisk En/Di; Int Graphics En/Di; IntEthernet En/Di; IntCOM1 En/Di; IntCOM2 En/Di; IntLPT En/Di; Int Keyboard En/Di; IntCPU En/Di; Int Misc En/Di; } Note: Indicates that Local Device Monitoring is Enabled or Disabled </pre>	<pre> { IntFloppy IO/Intr/VxD; Int 1 larddisk IO/Intr/VxD; Int Graphics IO/Intr/VxD; IntEthernet IO/Intr/VxD; IntCOM1 IO/Intr/VxD; IntCOM2 IO/Intr/VxD; IntLPT IO/Intr/VxD; Int Keyboard IO/Intr/VxD; IntCPU IO/Intr/VxD; Int Misc IO/Intr/VxD; } Note: Indicates the Local Device Events that are Monitored </pre>	<pre> { IntFloppy On/Off; Int 1 larddisk On/Off; Ont Graphics On/Off; IntEthernet On/Off; IntCOM1 On/Off; IntCOM2 On/Off; IntLPT On/Off; Int Keyboard On/Off; IntCPU On/Off; Int Misc On/Off; } Note: Indicates the Local Device Power On/Off Status </pre>

TABLE 2

Struct Local_Activity	Struct Local_Dint.Num	Struct Local_D.I.O.Range
<pre> { IntFloppy Yes/No; IntHarddisk Yes/No; IntGraphics Yes/No; IntEthernet Yes/No; IntCOM1 Yes/No; IntCOM2 Yes/No; Int LPT Yes/No; Int Keyboard Yes /No; Int CPU Yes/No; Int Misc Yes/No; } Note: Indicates that Local Device is Active/Inactive since last sampling by PM manager </pre>	<pre> { IntFloppy Num; IntHarddisk Num; IntGraphics Num; IntEthernet Num; IntCOM1 Num; IntCOM2 Num; Int LPT Num; Int Keyboard Num; Int CPU Num; Int Misc Num; } Note: Indicates the Local Device Interrupt Number to be monitored as Event </pre>	<pre> { IntFloppy IO Range; IntHarddisk IO Range; IntGraphics IO Range; IntEthernet IO Range; IntCOM IO Range; IntCOM2 IO Range; Int LPY IO Range; Int Keyboard IO Range; Int CPU IO Range; Int Misc IO Range; } Note: Indicates the Local Device IO Range as 16-bit Start/End Addr. Pair </pre>

55

Table 3 below illustrates the PMVxD initialization data structures for global events.

TABLE 3

StructGlobal_Sys.Events	StructSys_Break.Events	StructSuspend_Status
<pre> { IntAPM_Msg. En/Di; IntNMI En/Di; IntRING En/Di; } </pre>	<pre> { IntAPM_Msg. En/Di; IntNMI En/Di; IntRING En/Di; } </pre>	<pre> { IntLocal_Standby On/Off; IntGlobal_Standby On/Off; IntFully_On On/Off; } </pre>

TABLE 3-continued

StructGlobal_Sys.Events	StructSys_Break.Events	StructSuspend_Status
IntIRQ<0:15> En/Di; Int Misc En/Di; }	IntIRQ<0:15> En/Di; Int Misc En/Di; }	IntHibernation On/Off; Int Misc On/Off; }
Note: Indicates that System Events Monitoring is Enabled or Disabled	Note: Indicates that System Events Enabled as Break Events	Note: Indicates the System Power Miser Mode Status

In one embodiment, a user interface for programming the power management features is available at an end-user level. Such an interface enables the user to specify data for the data structures shown above. Note that in one embodiment, this 15 may be invisible to the user

In one embodiment, hardware is manipulated directly, for example, when the VxD and VMM service calls provided by Windows are limited. For instance, in one embodiment, it may be necessary to disable the system timer interrupt 20 during Standby mode.

In one embodiment, in order to trap I/O address ranges which are virtualized by other VxDs, interrupt chaining or use of a device driver hook handler is required. Power Managed Hardware and An Exemplary Computer 25 System

The various components of a system that may be power managed include, but are not limited to, the CPU, generic I/O controllers (e.g., graphics, floppy, hard disk, keyboard, etc.), serial and parallel interfaces and their associated 30 devices (e.g., modem, mouse, trackball, trackpad, printers, LAN), LAN interfaces, and DRAM memory systems. An exemplary computer system is described in FIG. 7 below.

Referring to FIG. 7, one embodiment of the computer system of the present invention is implemented is shown as 35 700. Computer system 700 comprises a bus or other communication means 701 for communicating information, and a processor 702 coupled with bus 701 for processing information. Processor 702 includes, but is not limited to microprocessors such as the Intel™ Architecture Microprocessor, 40 manufactured by Intel Corporation of Santa Clara, Calif., the corporate assignee of the present invention, PowerPC™, Alpha™, etc.

System 700 further comprises a random access memory (RAM) or other dynamic storage device 704 (referred to as 45 main memory), coupled to bus 701 for storing information and instructions to be executed by processor 702. Main memory 704 also may be used for storing temporary variables or other intermediate information during execution of instructions by processor 702. Computer system 700 also 50 comprises a read only memory (ROM) and/or other static storage device 706 coupled to bus 101 for storing static information and instructions for processor 702, and a data storage device 707 such as a magnetic disk or optical disk and its corresponding disk drive. Data storage device 707 is 55 coupled to bus 701 for storing information and instructions.

Computer system 700 may further be coupled to a display device 721, such as a cathode ray tube (CRT) or liquid crystal display (LCD) coupled to bus 701 for displaying 60 information to a computer user. An alphanumeric input device 722, including alphanumeric and other keys, may also be coupled to bus 701 for communicating information and command selections to processor 702. An additional user input device is cursor control 723, such as a mouse, a trackball, stylus, or cursor direction keys, coupled to bus 701 65 for communicating direction information and command selections to processor 702, and for controlling cursor move-

ment on display 721. Another device which may be coupled to bus 701 is hard copy device 724 which may be used for printing instructions, data, or other information on a medium such as paper, film, or similar types of media. Furthermore, a sound recording and playback device, such as a speaker and microphone may optionally be coupled to bus 701 for interfacing with computer system 700.

Note that any or all of the components of system 700 and associated hardware may be used, however, it can be appreciated that any type of configuration of the system may be used for various purposes as the user requires.

The PMVxD of the present invention requires none or very minimal hardware support to achieve comparable or better level of power conservation, relative to the prior art dedicated hardware (SMM) power management techniques discussed above.

The present invention provides power management using less hardware. Because less hardware is required, there is less power consumption. Thus, the power management provided by the present invention is a power saver.

As opposed to the prior art, the present invention allows more room for expansion without any hardware penalty. Also the I/O devices used in the present invention are not required to use standard I/O addresses over the life of the system.

The power management mechanism of the present invention is closely coupled to the operating system, such that each is aware of the other's actions. This allows the present invention to avoid system crashes (e.g., when the power management software interrupts and takes control from the operating system when it is executing a critical section of code). In other words, the PMVxD of the present invention eliminates system crash problems by eliminating blind spots. Also, the close coupling allows the PMVxD to be aware of dynamic changes in the I/O device state changes. Furthermore, the present invention does not have to virtualize I/O ports. There is no difficulty for the PMVxD to detect it as such. Thus, the present invention prevents monitoring and trapping on invalid I/O device addresses and the generation of improper events in the system.

When I/O device addresses do change dynamically during the life time of the current system boot, the present invention provides an easy environment in which to update and reorganize it by simply modifying the data structures and their I/O address ranges.

Moreover, the present invention operates with existing software power management techniques (e.g., APM 1.1 spec.) that assumes that applications and the associated device drivers in the system be APM-aware to monitor and control power management. Thus, the PMVxD of the present invention power manages a system which has traditional applications and device drivers, one of which is not power-aware. The PMVxD makes the non-power-aware Windows applications and device drivers "virtually power aware".

The present invention may be extended to operating systems like IBM OS/2, Microsoft® Windows NT, Unix, etc.

Although the above description focused on notebook computer system power management, the present invention may be applied to desktop and home computer systems as well. Alternatively, the present invention may be used to power manage PCMCIA hardware and for thermal control of future generation CPU's.

Note that for Real Mode DOS or Standard Mode Windows environments, it may be appropriate to use POWER-EXE for real mode power management.

Thus, a method and apparatus for reducing power consumption in a computer system using virtual device drivers has been described.

I claim:

1. A computer system comprising:

a bus; 15  
at least one memory coupled to the bus for storing data, including an operating system; and  
a central processing unit (CPU) coupled to the bus running the operating system with a virtual device driver (VxD), wherein the virtual device driver performs device idle detection using one or more events timers indicating the activity level of at least one local device, and further wherein the virtual device driver places idle local devices in a reduced power consumption state when associated events timers indicate that no activity has occurred for a predetermined period of time. 20 25

2. The computer system defined in claim 1 wherein the virtual device driver performs system idle detection.

3. The computer system defined in claim 1 wherein the virtual device driver comprises I/O trapping capabilities to perform idle detection. 30

4. The computer system defined in claim 1 wherein the virtual device driver comprises a VxD trap handler to perform idle detection. 35

5. The computer system defined in claim 1 wherein the virtual device driver comprises a chained-interrupt trap handler to perform idle detection.

6. The computer system defined in claim 1 wherein the memory stores data structures indicating enabled local devices being monitored by the device driver. 40

7. The computer system defined in claim 1 wherein the memory stores data structures indicating events being monitored by the device driver.

8. The computer system defined in claim 1 wherein the memory stores data structures indicating I/O address ranges for local devices. 45

9. The computer system defined in claim 1 wherein the memory stores data structures indicating activity level of local devices to the device driver.

10. The computer system defined in claim 1 wherein the memory stores data structures indicating power management states into which the device driver may place the computer system. 50

11. A computer system comprising:

a bus; 55  
a central processing unit (CPU) coupled to the bus running an operating system and at least one power-unaware application, wherein the operating system has a virtual device driver performing device idle detection using one or more events timers indicating the activity level of at least one local device, and further wherein the virtual device driver places idle local devices in a reduced power consumption state when associated events timers indicate that no activity has occurred for a predetermined period of time transparent to said at least one power-unaware application. 60 65



12. The computer system defined in claim 11 wherein the virtual device driver performs system idle detection.

13. The computer system defined in claim 11 wherein the virtual device driver comprises I/O trapping capabilities to perform idle detection.

14. The computer system defined in claim 11 wherein the virtual device driver comprises a VxD trap handler to perform idle detection.

15. The computer system defined in claim 11 wherein the virtual device driver comprises a chained-interrupt trap handler to perform idle detection.

16. The computer system defined in claim 11 wherein the memory stores data structures indicating enabled local devices being monitored by the virtual device driver.

17. The computer system defined in claim 11 wherein the memory stores data structures indicating events being monitored by the virtual device driver.

18. The computer system defined in claim 11 wherein the memory stores data structures indicating I/O address ranges for local devices.

19. The computer system defined in claim 11 wherein the memory stores data structures indicating activity level of local devices to the virtual device driver.

20. The computer system defined in claim 11 wherein the memory stores data structures indicating power management states into which the virtual device driver may place the computer system.

21. A computer system comprising:

at least one bus;

a memory coupled to said at least one bus;

a device coupled to said at least one bus;

a processor coupled to said at least one bus, wherein the processor is configured to execute a virtual device driver to control placement of said device into a reduced power consumption state.

22. The computer system defined in claim 21 wherein the virtual device driver performs idle detection for the device.

23. The computer system defined in claim 21 wherein the virtual device driver performs idle detection for the device using at least one event timer indicating the activity level of the device.

24. The computer system defined in claim 23 wherein the virtual device driver places the device in a reduced power consumption state when said at least one events timer indicates that no activity has occurred for a predetermined period of time.

25. The computer system defined in claim 21 wherein the processor runs at least one power-unaware application and the virtual device driver places the device in the reduced power consumption state transparent to said at least one power-unaware application.

26. The computer system defined in claim 21 wherein the memory stores data structures indicating enabled local devices being monitored by the virtual device driver, events being monitored by the virtual device driver, I/O address ranges for local devices, and activity level of local devices to the virtual device driver.

27. The computer system defined in claim 21 wherein the virtual device driver comprises a VxD trap handler to perform idle detection.

28. The computer system defined in claim 21 wherein the device comprises an I/O device.

29. A method for controlling an input/output (I/O) device, said method comprising the steps of:

executing a virtual device driver;

monitoring activity of the I/O device;

detecting the I/O device being inactive for a predetermined period of time; and

the virtual device driver placing the I/O device in a reduced power consumption state in response to the I/O device being detected as inactive.

30. The method defined in claim 29 further comprising the step of initializing, at boot-up time, a plurality of data structures associated with the virtual device driver.

31. The method defined in claim 29 wherein the step of monitoring comprises the virtual device driver monitoring activity of the I/O device at the occurrence of a system timer interrupt.

32. The method defined in claim 29 further comprising the step of varying the predetermined period of time.

\* \* \* \* \*

36. An article comprising a plurality of computer instructions, which when executed by a computer system, causes the computer system to perform the steps of:

monitoring activity of a device;

detecting the device being inactive for a predetermined period of time;

placing the device in a reduced power consumption state in response to the device being detected as inactive; and

managing said device to allow more than one application to use the device.

37. The article of claim 36, wherein the plurality of instructions is a device driver.

38. The article of claim 37, wherein the device is an Input/Output device.

39. The article of claim 38, wherein execution of the plurality of instructions, causes the computer system to further perform the steps of:

checking a status of an event timer at regular intervals.

40. The article of claim 39, wherein the plurality of instructions includes at least one idle timer.

41. A method for managing devices within a computer system, said method comprising of:

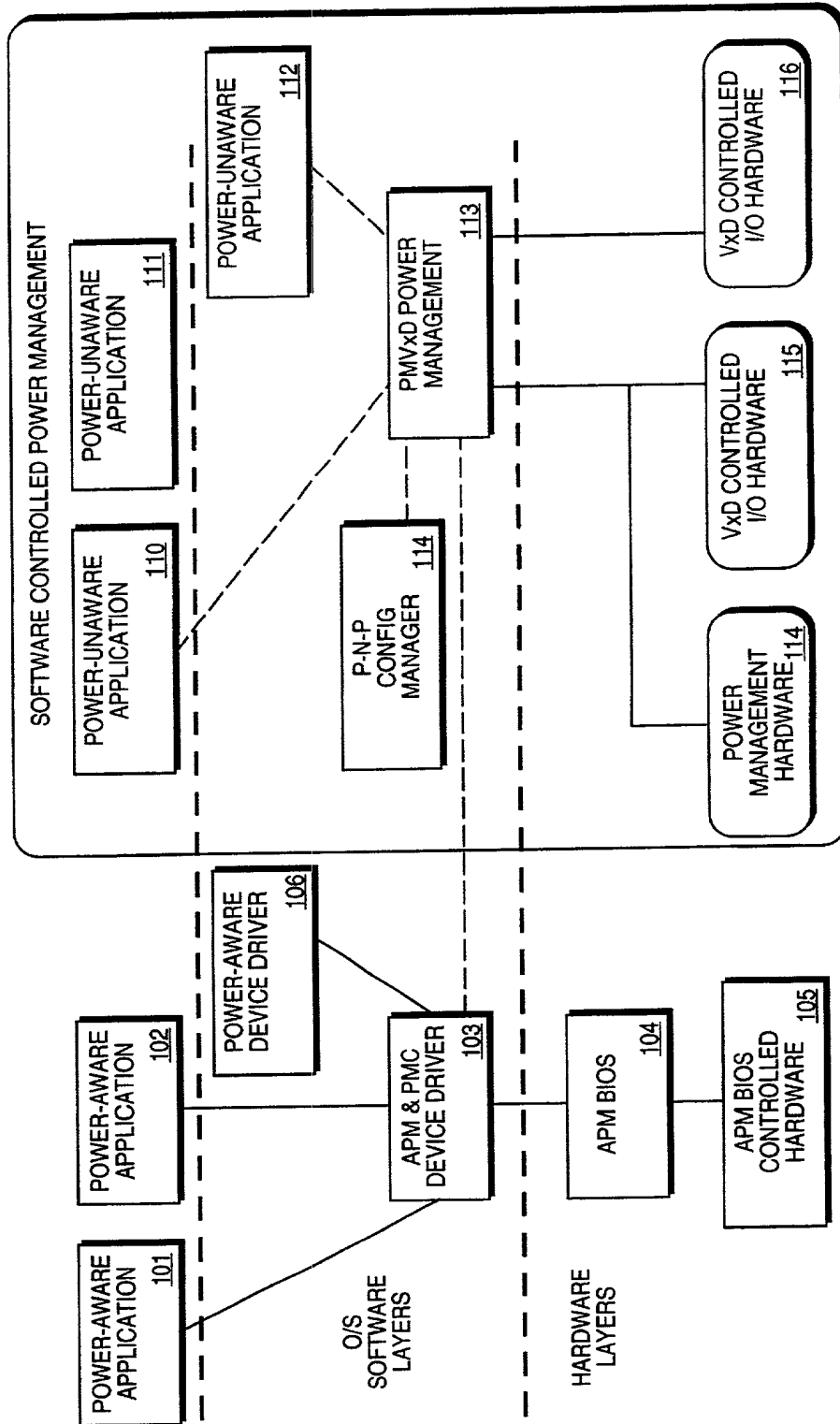
monitoring activity of a device;

detecting the device being inactive for a predetermined period of time;

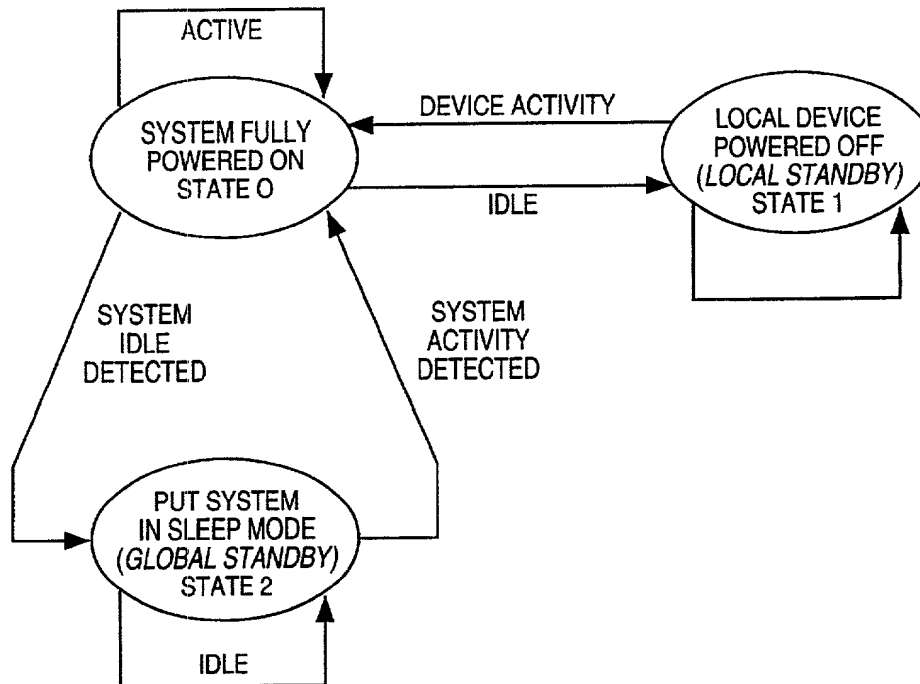
placing the device in a reduced power consumption state in response to the device being detected as inactive; and

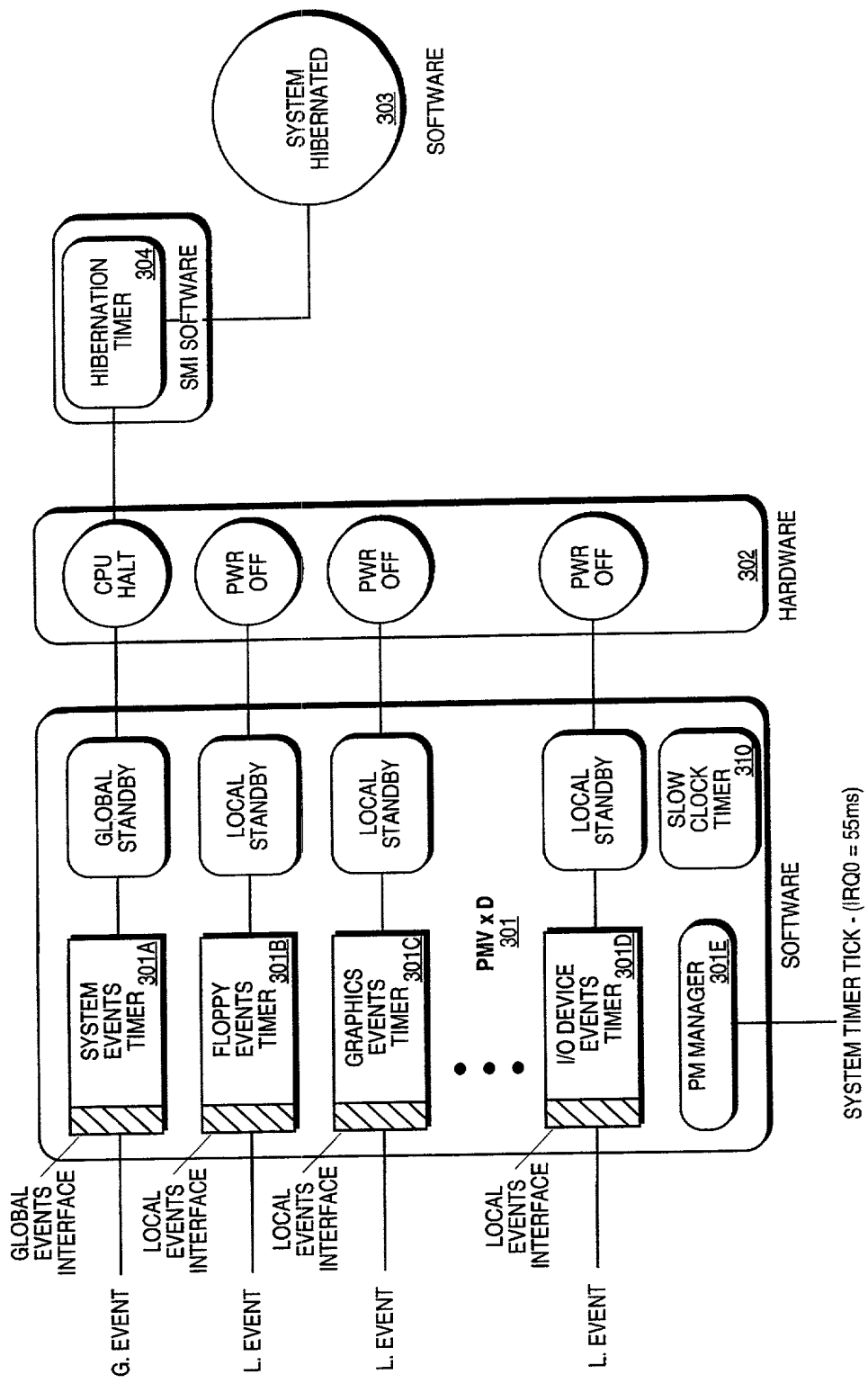
managing the device to allow more than one application to use the device.

42. The method of claim 41, wherein the method is performed by a device driver.
43. The method of claim 42, wherein the device is an Input/Output device.
44. The method of claim 43, wherein the method further includes the step of:  
checking a status of an event timer at regular intervals.
45. The method of claim 44, wherein the device driver includes at least one idle timer.
46. A computer system comprising:  
a bus;  
a memory coupled to the bus;  
a first device coupled to the bus;  
a processor coupled to the bus; and  
a second device having stored thereon a set of instructions, which when executed by said processors, manage the first device to allow more than one application to use the first device, and monitor activity of the first device, and in response to said first device being inactive for a predetermined period of time, place the first device in a reduced power consumption state.
47. The computer system of claim 46, wherein the set of instructions stored on the second device is a device driver.
48. The computer system of claim 47, wherein the first device is an Input/Output device.
49. The computer system of claim 48, wherein the device driver checks a status of an event timer at regular intervals.
50. The computer system of claim 49, wherein the device driver includes at least one idle timer.



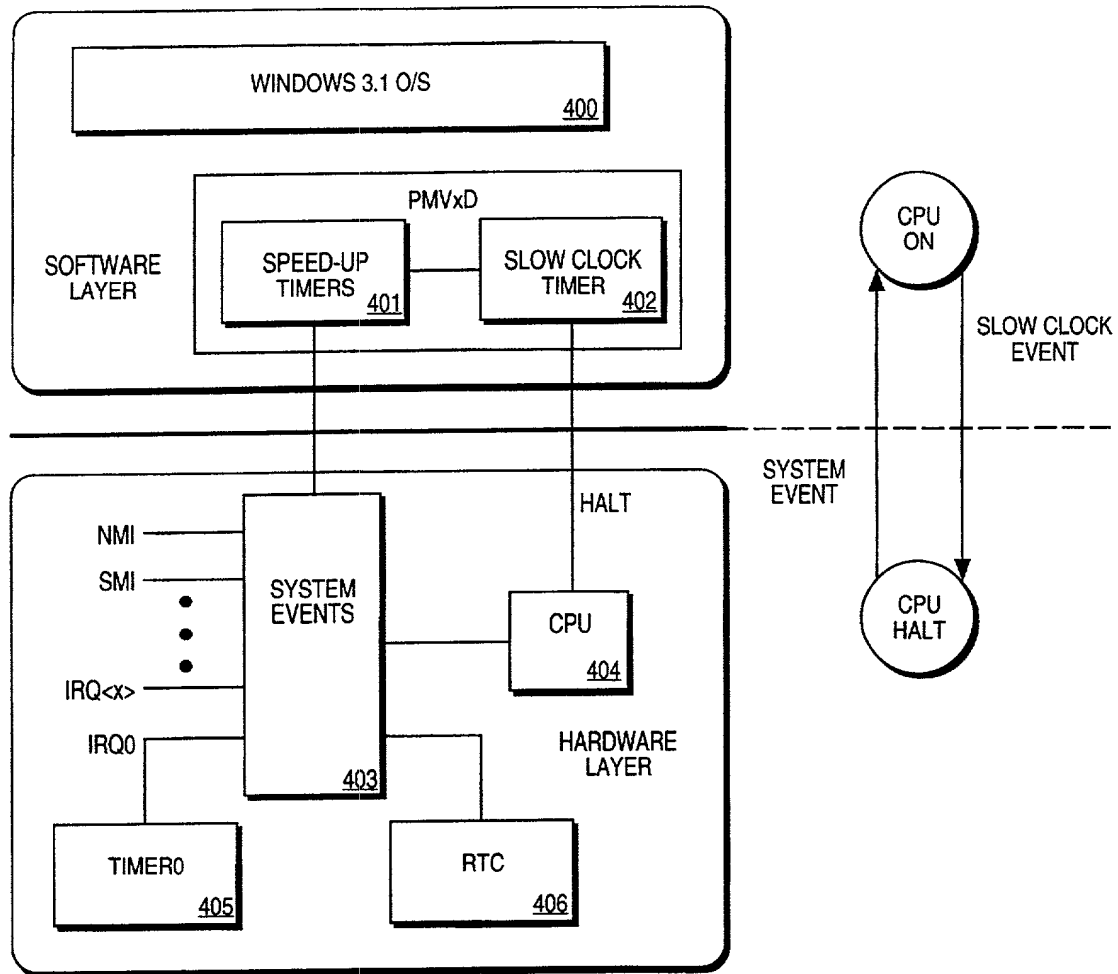
**FIG. 1**

**FIG. 2**



**FIG. 3**





**FIG. 4**

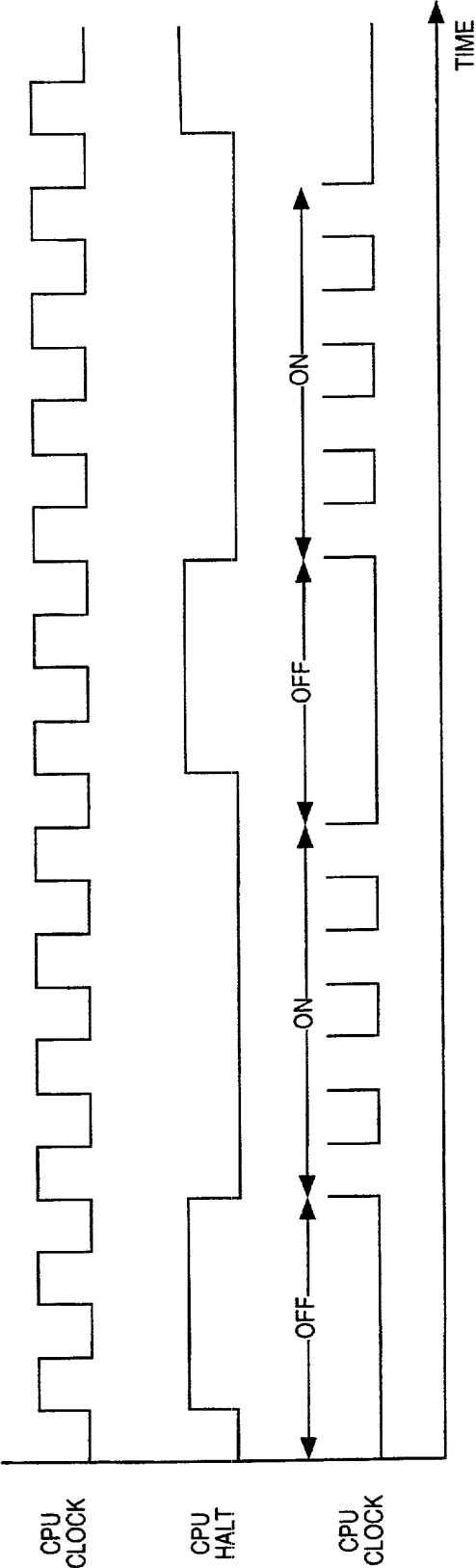


FIG. 5

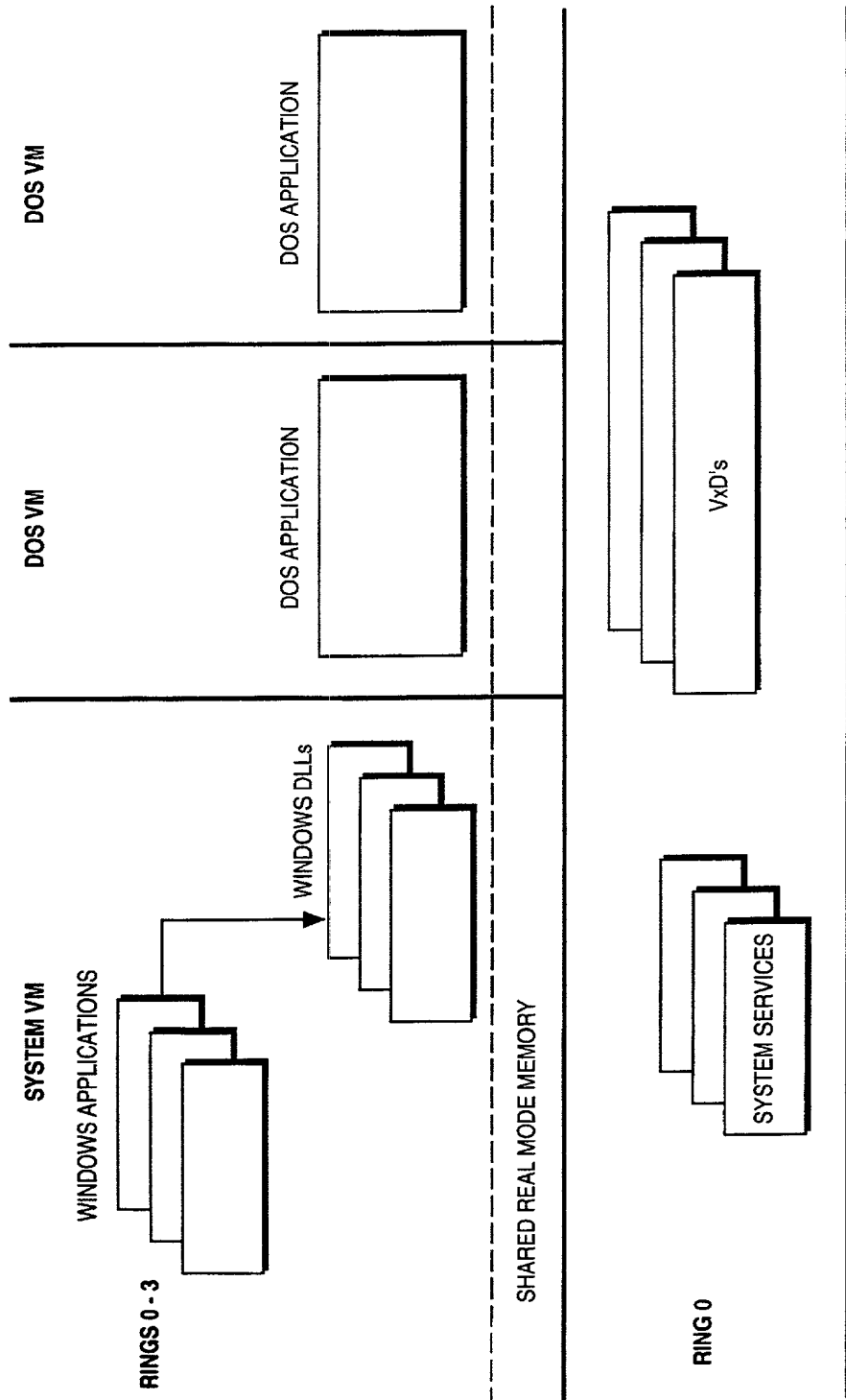


FIG. 6

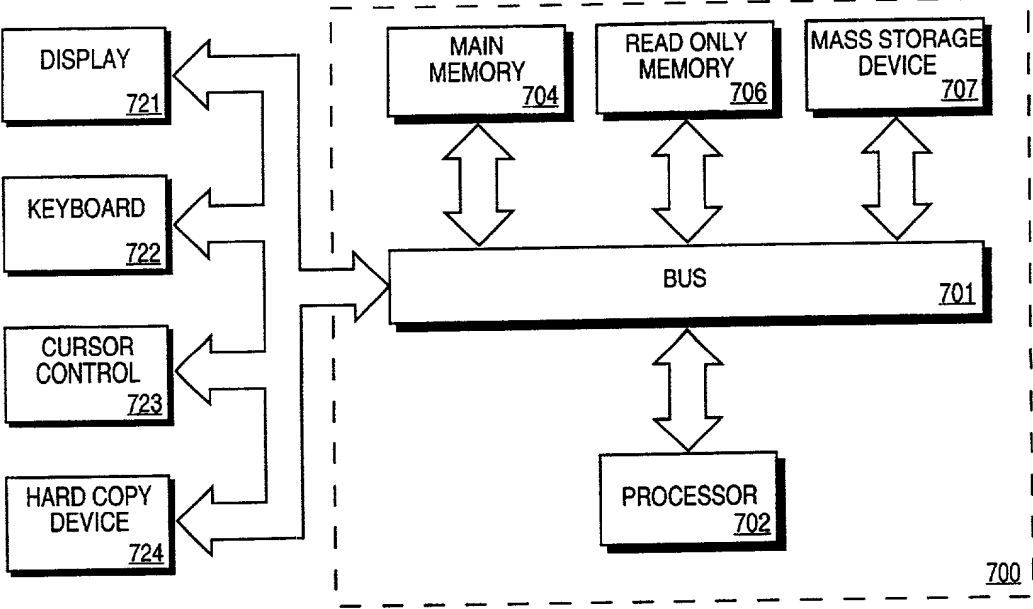


FIG. 7

DECLARATION AND POWER OF ATTORNEY FOR REISSUE PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below, next to my name.

I believe I am the original, first, and sole inventor (if only one name is listed below) or an original, first, and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

Method And Apparatus For Reducing Power Consumption In A Computer System Using Virtual Device Drivers

the specification of which is attached hereto and was issued as U.S. Patent No. 5,590,342 (the "original patent") from application number 08/346,040, filed November 29, 1994 (the "original application").

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claim(s), as amended by any amendment referred to above. I do not know and do not believe that the claimed invention was ever known or used in the United States of America before my invention thereof, or patented or described in any printed publication in any country before my invention thereof or more than one year prior to the original application, that the same was not in public use or on sale in the United States of America more than one year prior to the original application, and that the invention has not been patented or made the subject of an inventor's certificate issued before the date of the original application in any country foreign to the United States of America on an application filed by me or my legal representatives or assigns more than twelve months (for a utility patent application) or six months (for a design patent application) prior to the original application.

I acknowledge the duty to disclose all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, Section 119(a)-(d), of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

<u>Prior Foreign Application(s)</u>			<u>Priority Claimed</u>	
<u>(Number)</u>	<u>(Country)</u>	<u>(Day/Month/Year Filed)</u>	<u>Yes</u>	<u>No</u>
<u>(Number)</u>	<u>(Country)</u>	<u>(Day/Month/Year Filed)</u>	<u>Yes</u>	<u>No</u>

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Assistant Commissioner for Patents  
Washington, D.C. 20231

REISSUE APPLICATION BY THE INVENTOR, OFFER TO SURRENDER  
(37 CFR 1.178)

The undersigned applicants of the accompanying reissue application for the  
reissue of letters patents for the improvement in Method And Apparatus  
For Reducing Power Consumption In A Computer System  
Using Virtual Device Drivers

Patent No. 5,590,342 granted to him/her on December 31, 1996 of which  
he/she is the sole owner  
X Intel Corporation  
now sole owner by assignment, and on whose behalf and  
with whose assent the accompanying application is made

hereby offers to surrender said letters patents.

Filed herewith is an  
X abstract of title, duly certified  
order for a title report as required in such applications.

Dated: \_\_\_\_\_  
Suresh K. Marisetty

ASSENT OF ASSIGNEE TO REISSUE

The undersigned assignee of the entire interest in the above-mentioned  
letters patents, hereby assents to the accompanying application.

Respectfully submitted,  
BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

Dated: \_\_\_\_\_  
Carl L. Silverman  
Director, Intellectual Property  
Intel Corporation

"Express-Mail" mailing label number: E103414029145

Date of Deposit: December 31, 1998

I hereby certify that I am causing this paper or fee to be  
deposited with the United States Postal Service "Express Mail Post  
Office to Addressee" service on the date indicated above and  
that this paper or fee has been addressed to the Assistant  
Commissioner for Patents, Washington, D.C. 20231

Lisa Kaiser  
(Typed or printed name of person mailing paper or fee)

Lisa Kaiser  
(Signature of person mailing paper or fee)

12/31/98  
(Date signed)

I hereby claim the benefit under title 35, United States Code, Section 119(e) of any United States provisional application(s) listed below

(Application Number)

Filing Date

I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application:

(Application Number)

Filing Date

(Status -- patented,  
pending, abandoned)

(Application Number)

Filing Date

(Status -- patented,  
pending, abandoned)

I hereby appoint Aloysius T. C. AuYeung, Reg. No. 35,432; William Thomas Babbitt, Reg. No. 39,591; Jordan Michael Becker, Reg. No. 39,602; Bradley J. Berezna, Reg. No. 33,474; Michael A. Bernadacou, Reg. No. 35,934; Roger W. Blakely, Jr., Reg. No. 25,831; Gregory D. Caldwell, Reg. No. 39,926; Kent M. Chen, Reg. No. 39,630; Lawrence M. Cho, Reg. No. 39,942; Thomas M. Coester, Reg. No. 39,637; Roland B. Cortes, Reg. No. 39,152; William Donald Davis, Reg. No. 38,428; Michael Anthony DeSanctis, Reg. No. 39,957; Daniel M. De Vos, Reg. No. 37,813; Karen L. Feisthamel, Reg. No. 40,264; James Y. Go, Reg. No. P-40,621; Tarek N. Fahmi, Reg. No. P-41,402; David R. Halvorson, Reg. No. 33,395; Eric Ho, Reg. No. 39,711; George W. Hoover II, Reg. No. 32,992; Eric S. Hyman, Reg. No. 30,139; Dag H. Johansen, Reg. No. 36,172; Stephen L. King, Reg. No. 19,180; Dolly M. Lee, Reg. No. 39,742; Michael J. Mallie, Reg. No. 36,591; Kimberley G. Nobles, Reg. No. 38,255; Ronald W. Reagin, Reg. No. 20,340; James H. Salter, Reg. No. 35,668; William W. Schaal, Reg. No. 39,018; James C. Scheller, Reg. No. 31,195; Charles E. Shemwell, Reg. No. 40,171; Maria McCormack Sobrino, Reg. No. 31,639; Stanley W. Sokoloff, Reg. No. 25,128; Allan T. Sponseller, Reg. No. 38,318; Steven R. Sponseller, Reg. No. 39,384; Edwin H. Taylor, Reg. No. 25,129; Lester J. Vincent, Reg. No. 31,460; John Patrick Ward, Reg. No. 40,216; Ben J. Yorks, Reg. No. 33,609; and Norman Zafman, Reg. No. 26,250; my attorneys; and Robert Andrew Diehl, Reg. No. P-40,992; Sharmini Nathan Green, Reg. No. P-41,410; Thomas A. Hassing, Reg. No. 36,159; Edwin A. Sloane, Reg. No. 34,728; and Judith A. Szepesi, Reg. No. 39,393; my patent agents, of BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP, with offices located at 12400 Wilshire Boulevard, 7th Floor, Los Angeles, California 90025, telephone (310) 207-3800, and Joseph R. Bond, Reg. No. 36,458; Richard C. Calderwood, Reg. No. 35,468; Sean Fitzgerald, Reg. No. 32,027; Naomi Obinata, Reg. No. 39,320; Thomas C. Reynolds, Reg. No. 32,488; Howard A. Skaist, Reg. No. 36,008; and Raymond J. Werner, Reg. No. 34,752; my patent attorneys, of INTEL CORPORATION with full power of substitution and revocation, to prosecute this application and to transact all business in the Patent and Trademark Office connected herewith.

I verily believe the original patent to be wholly or partially inoperative by reason that the patent claims less than I had a right to claim in the patent. The claim contains excess limitations not necessary for patentability. For example, the claim contains excess limitations not necessary for distinguishing over the prior art. Also, the claim fails to cover embodiments of the invention as claimed in the above-identified reissue application. The error arose during the drafting of the original application and during subsequent amendments in connection with the prosecution of the original application which resulted in the issuance of the patent. The attorney prosecuting the original application failed to appreciate the scope of the invention, and thus limited the claims as indicated above. The error arose without any deceptive intention on my part. I further acknowledge my duty to disclose information which is material to the examination of the application under 37 CFR § 1.56.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Sole/First Inventor Suresh K. Marisetty

Inventor's Signature \_\_\_\_\_ Date \_\_\_\_\_

Residence San Jose, California Citizenship India  
(City, State) (Country)

Post Office Address 1662 Lederer Circle  
San Jose, California 95131

Title 37, Code of Federal Regulations, Section 1.56  
Duty to Disclose Information Material to Patentability

(a) A patent by its very nature is affected with a public interest. The public interest is best served, and the most effective patent examination occurs when, at the time an application is being examined, the Office is aware of and evaluates the teachings of all information material to patentability. Each individual associated with the filing and prosecution of a patent application has a duty of candor and good faith in dealing with the Office, which includes a duty to disclose to the Office all information known to that individual to be material to patentability as defined in this section. The duty to disclosure information exists with respect to each pending claim until the claim is cancelled or withdrawn from consideration, or the application becomes abandoned. Information material to the patentability of a claim that is cancelled or withdrawn from consideration need not be submitted if the information is not material to the patentability of any claim remaining under consideration in the application. There is no duty to submit information



which is not material to the patentability of any existing claim. The duty to disclose all information known to be material to patentability is deemed to be satisfied if all information known to be material to patentability of any claim issued in a patent was cited by the Office or submitted to the Office in the manner prescribed by §§1.97(b)-(d) and 1.98. However, no patent will be granted on an application in connection with which fraud on the Office was practiced or attempted or the duty of disclosure was violated through bad faith or intentional misconduct. The Office encourages applicants to carefully examine:

(1) Prior art cited in search reports of a foreign patent office in a counterpart application, and

(2) The closest information over which individuals associated with the filing or prosecution of a patent application believe any pending claim patentably defines, to make sure that any material information contained therein is disclosed to the Office.

(b) Under this section, information is material to patentability when it is not cumulative to information already of record or being made or record in the application, and

(1) It establishes, by itself or in combination with other information, a prima facie case of unpatentability of a claim; or

(2) It refutes, or is inconsistent with, a position the applicant takes in:

(i) Opposing an argument of unpatentability relied on by the Office, or

(ii) Asserting an argument of patentability.

A prima facie case of unpatentability is established when the information compels a conclusion that a claim is unpatentable under the preponderance of evidence, burden-of-proof standard, giving each term in the claim its broadest reasonable construction consistent with the specification, and before any consideration is given to evidence which may be submitted in an attempt to establish a contrary conclusion of patentability.

(c) Individuals associated with the filing or prosecution of a patent application within the meaning of this section are:

(1) Each inventor named in the application;

(2) Each attorney or agent who prepares or prosecutes the application; and

(3) Every other person who is substantively involved in the preparation or prosecution of the application and who is associated with the inventor, with the assignee or with anyone to whom there is an obligation to assign the application.

(d) Individuals other than the attorney, agent or inventor may comply with this section by disclosing information to the attorney, agent, or inventor.